08957512 . 102457

APPENDIX E

"EXPRESS MAIL" Mailing Label Number EI267842785US

Date of Deposit October 24, 1997

I hereby certify under 37 CFR 1.10 that this correspondence is being deposited with the United States Postal Service as **"Express Mail Post Office To Addressee"** with sufficient postage on the date indicated above and is addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231.

[illegible]

Tina Grimstead-Campbell

APPENDIX E

Example Loading And Execution Control Program

```
public class  Bootstrap {

  // Constants used throughout the program
  static final byte BUFFER_LENGTH          = 32;
  static final byte ACK_SIZE               = (byte)1;
  static final byte ACK_CODE               = (byte)0;
  static final byte OS_HEADER_SIZE         = (byte)0x10;
  static final byte GPOS_CREATE_FILE       = (byte)0xE0;

  static final byte ST_INVALID_CLASS       = (byte)0xC0;
  static final byte ST_INVALID_PARAMETER   = (byte)0xA0;
  static final byte ST_INS_NOT_SUPPORTED   = (byte)0xB0;
  static final byte ST_SUCCESS             = (byte)0x00;

  static final byte ISO_COMMAND_LENGTH     = (byte)5;
  static final byte ISO_READ_BINARY        = (byte)0xB0;
  static final byte ISO_UPDATE_BINARY      = (byte)0xD6;
  static final byte ISO_INIT_APPLICATION   = (byte)0xF2;
  static final byte ISO_VERIFY_KEY         = (byte)0x2A;
  static final byte ISO_SELECT_FILE        = (byte)0xA4;

  static final byte ISO_CLASS              = (byte)0xC0;
  static final byte ISO_APP_CLASS          = (byte)0xF0;


  public static void main () {

    byte pbuffer[] = new byte[ISO_COMMAND_LENGTH];
    byte dbuffer[] = new byte[BUFFER_LENGTH];
    byte ackByte[] = new byte[ACK_SIZE];
    //short fileId;
    short offset;
    byte bReturnStatus;

    // Initialize Communications
    _OS.SendATR();

    do {
        // Retrieve the command header
        _OS.GetMessage(pbuffer, ISO_COMMAND_LENGTH, ACK_CODE);

        // Verify class of the message - Only ISO + Application
        if ((pbuffer[0] != ISO_APP_CLASS)
         && (pbuffer[0] != ISO_CLASS)) {
            _OS.SendStatus(ST_INVALID_CLASS);
        }
        else {
          // go through the switch
          // Send the acknowledge code

          // Verify if data length too large
          if (pbuffer[4] > BUFFER_LENGTH) {
            bReturnStatus = ST_INVALID_PARAMETER;
          }
          else
          {
            switch (pbuffer[1]) {
            case ISO_SELECT_FILE:
                // we always assume that length is 2
                if (pbuffer[4] != 2) {
                    bReturnStatus = ST_INVALID_PARAMETER;
                }
                else
                {
                    // get the fileId(offset) in the data buffer
                    _OS.GetMessage(dbuffer, (byte)2, pbuffer[1]);
                    // cast dbuffer[0..1] into a short
                    offset = (short) ((dbuffer[0] << 8) | (dbuffer[1] & 0x00FF));
                    bReturnStatus = _OS.SelectFile(offset);
                }
                break;

            case ISO_VERIFY_KEY:
                // Get the Key from the terminal
                _OS.GetMessage(dbuffer, pbuffer[4], pbuffer[1]);

                bReturnStatus = _OS.VerifyKey(pbuffer[3],
                                              dbuffer,
                                              pbuffer[4]);
                break;

            case ISO_INIT_APPLICATION:
                // Should send the id of a valid program file
                _OS.GetMessage(dbuffer, (byte)1, pbuffer[1]);
                // compute fileId(offset) from pbuffer[2..3] via casting
                offset = (short) ((pbuffer[2] << 8) | (pbuffer[3] & 0x00FF));
                bReturnStatus = _OS.Execute(offset,
                                            dbuffer[0]);
                break;
            case GPOS_CREATE_FILE:
                if (pbuffer[4] != OS_HEADER_SIZE) {
                    bReturnStatus = ST_INVALID_PARAMETER;
                    break;
                }
                // Receive The data
                _OS.GetMessage(dbuffer, pbuffer[4], pbuffer[1]);
                bReturnStatus = _OS.CreateFile(dbuffer);
                break;

            case ISO_UPDATE_BINARY:
                _OS.GetMessage(dbuffer, pbuffer[4], pbuffer[1]);
                // compute offset from pbuffer[2..3] via casting
                offset = (short) ((pbuffer[2] << 8) | (pbuffer[3] & 0x00FF));
                // assumes that a file is already selected
                bReturnStatus = _OS.WriteBinaryFile (offset,
                                                     pbuffer[4],
                                                     dbuffer);
                break;
            case ISO_READ_BINARY:
                // compute offset from pbuffer[2..3] via casting
                offset = (short) ((pbuffer[2] << 8) | (pbuffer[3] & 0x00FF));
                // assumes that a file is already selected
                bReturnStatus = _OS.ReadBinaryFile (offset,
                                                    pbuffer[4],
                                                    dbuffer);
                // Send the data if successful
                ackByte[0] = pbuffer[1];
                if (bReturnStatus == ST_SUCCESS) {
                    _OS.SendMessage(ackByte, ACK_SIZE);
                    _OS.SendMessage(dbuffer, pbuffer[4]);
                }
                break;
            default:
                bReturnStatus = ST_INS_NOT_SUPPORTED;
            }
          }
        _OS.SendStatus(bReturnStatus);
      }
    }
    while (true);
  }
}
```

08957512.102497

APPENDIX F

"EXPRESS MAIL" Mailing Label Number EI267842785US

Date of Deposit October 24, 1997

I hereby certify under 37 CFR 1.10 that this correspondence is being deposited with the United States Postal Service as **"Express Mail Post Office To Addressee"** with sufficient postage on the date indicated above and is addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231.

[illegible]

Tina Grimstead-Campbell

APPENDIX F

Methods For Accessing Card Operating System Capabilities In The Preferred Embodiment

```
public class _OS {

  static native byte        SelectFile          (short  file_id);
  static native byte        SelectParent        ();
  static native byte        SelectCD            ();
  static native byte        SelectRoot          ();

  static native byte        CreateFile          (byte   file_hdr[]);
  static native byte        DeleteFile          (short  file_id);

  // General File Manipulation
  static native byte        ResetFile           ();
  static native byte        ReadByte            (byte   offset);
  static native short       ReadWord            (byte   offset);

  // Header Manipulation
  static native byte        GetFileInfo         (byte   file_hdr[]);

  // Binary File support
  static native byte        ReadBinaryFile      (short  offset,
                                                  byte   data_length,
                                                  byte   buffer[]);
  static native byte        WriteBinaryFile     (short  offset,
                                                  byte   data_length,
                                                  byte   buffer[]);

  // Record File support
  static native byte        SelectRecord        (byte   record_nb,
                                                  byte   mode);
  static native byte        NextRecord          ();
  static native byte        PreviousRecord      ();


  static native byte        ReadRecord          (byte   record_data[],
                                                  byte   record_nb,
                                                  byte   offset,
                                                  byte   length);
  static native byte        WriteRecord         (byte   buffer[],
                                                  byte   record_nb,
                                                  byte   offset,
                                                  byte   length);


  // Cyclic File Support
  static native byte        LastUpdatedRec      ();

  // Messaging Functions
  static native byte        GetMessage          (byte   buffer[],
                                                  byte   expected_length,
                                                  byte   ack_code);
  static native byte        SendMessage         (byte   buffer[],
                                                  byte   data_length);
  static native byte        SetSpeed            (byte   speed);

  // Identity Management
  static native byte        CheckAccess         (byte   ac_action);
  static native byte        VerifyKey           (byte key_number,
                                                  byte key_buffer[],
                                                  byte key_length);
  static native byte        VerifyCHV           (byte   CHV_number,
                                                  byte   CHV_buffer[],
                                                  byte   unblock_flag);
  static native byte        ModifyCHV           (byte   CHV_number,
                                                  byte   old_CHV_buffer[],
                                                  byte   new_CHV_buffer[],
                                                    byte   unblock_flag);
    static native byte        GetFileStatus        ();
    static native byte        SetFileStatus        (byte   file_status);

    static native byte        GrantSupervisorMode  ();
    static native byte        RevokeSupervisorMode();

    static native byte        SetFileACL           (byte   file_acl[]);
    static native byte        GetFileACL           (byte   file_acl[]);

    // File context manipulation
    static native void        InitFileStatus       ();
    static native void        BackupFileStatus     ();
    static native void        RestoreFileStatus    ();

    // Utilities
    static native byte        CompareBuffer        (byte   pattern_length,
                                                    byte   buffer_1[],
                                                    byte   buffer_2[]);
    static native short       AvailableMemory      ();
    static native void        ResetCard            (byte   mode);
    static native byte        SendATR              ();
    static native byte        SetDefaultATR        (byte   buffer[],
                                                    byte   length);
    static native byte        Execute              (short  file_id,
                                                    byte   flag);

    // Global state variable functions
    static native byte        GetIdentity          ();
    static native byte        GetRecordNb          ();
    static native short       GetApplicationId     ();
    static native byte        GetRecordLength      ();
    static native byte        GetFileType          ();
    static native short       GetFileLength        ();
    static native void        SendStatus           (byte status);

}
```

09057512.102497

F-2